Описание:

В этом задании необходимо провести анализ некоторого временного ряда и попробовать предсказать значения для последующих месяцев.

Постановка задачи:

1. Считать данные из training.xlsx. Проверить является ли ряд стационарным в широком смысле двумя способами:
    i. Провести визуальную оценку, отрисовав ряд и скользящую статистику(среднее, стандартное отклонение). Постройте график на котором будет отображен сам ряд и различные скользящие статистики.
    ii. Провести тест Дики - Фуллера.

Сделать выводы из полученных результатов. Оценить достоверность статистики.

2. Разложить временной ряд на тренд, сезональность, остаток в соответствии с аддитивной, мультипликативной моделями. Визуализировать их, оценить стационарность получившихся рядов, сделать выводы.

3. Проверить является ли временной ряд интегрированным порядка k. Если является, применить к нему модель ARIMA, подобрав необходимые параметры с помощью функции автокорреляции и функции частичной автокорреляции. Отобрать несколько моделей. Предсказать значения для тестовой выборки. Визуализировать их, посчитать r2 score для каждой из моделей. Произвести отбор наилучшей модели с помощью информационного критерия Акаике.

Теоретическая справка:

Временной ряд – это совокупность значений какого-либо показателя за несколько последовательных моментов или периодов времени.

Временные ряды содержат регулярную составляющую (обычно включающую несколько компонент) и случайный шум (ошибку), который затрудняет обнаружение регулярных компонент.

Большинство регулярных составляющих временных рядов принадлежит к двум классам: они являются либо трендом, либо сезонной составляющей. Тренд представляет собой общую систематическую линейную или нелинейную компоненту, которая может изменяться во времени. Сезонная составляющая - это периодически повторяющаяся компонента. Оба эти вида регулярных компонент часто присутствуют в ряде одновременно.

Таким образом, временной ряд может формироваться из трендовой (T), сезонной компоненты (S), а также случайной (E) компоненты.

Модели, где временной ряд представлен в виде суммы перечисленных компонентов называются *аддитивными*, если в виде произведения – *мультипликативными* моделями.

Общий вид аддитивной модели следующий:

$Y = T + S + E.$

Общий вид мультипликативной модели выглядит так:

Y = T·S·E.

Выбор одной из двух моделей осуществляется на основе анализа структуры сезонных колебаний. Если амплитуда колебаний приблизительно постоянна, строят аддитивную модель временного ряда, в которой значения сезонной компоненты предполагаются постоянными для различных циклов. Если амплитуда сезонных колебаний возрастает или уменьшается, строят мультипликативную модель временного ряда, которая ставит уровни ряда в зависимость от значений сезонной компоненты.

Анализ тренда

Если временные ряды содержат значительную ошибку, то первым шагом выделения тренда является сглаживание. Сглаживание всегда включает некоторый способ локального усреднения данных, при котором несистематические компоненты взаимно погашают друг друга. Самый общий метод сглаживания - *скользящее среднее,* в котором каждый член ряда заменяется средним *n* соседних членов, где *n* - ширина "окна".

Анализ сезонности

Сезонность – периодическая зависимость, которая может быть определена как корреляционная зависимость (статистическая зависимость между случайными величинами) порядка *k* между каждым *i*-м элементом ряда и *(i-k)*-м элементом. *k* обычно называют *сдвигом, запаздыванием.*

В анализе временных рядов *стационарные ряды* имеют постоянные по времени среднее и дисперсию, а ковариационная функция зависит только от сдвига. Если нарушается хотя бы одно из этих условий, то ряд является нестационарным.

Первые разности ряда $\Delta y_t = y_t - y_{t-1}$ – это изменение значения временного ряда за период [*t-1, t*].

Аналогично вводится *ряд вторых разностей* (ряд первых разностей от ряда первых разностей):

$$\Delta^2 y_t = \Delta(\Delta y_t) = y_t - 2y_{t-1} + y_{t-2}.$$

Ряд *k-х разностей*:

$$\Delta^k y_t = \Delta(\Delta^{k-1} y_t)$$

Авторегрессионная модель — модель временного ряда, в которой значения в данный момент линейно зависят от предыдущих значений этого же ряда.

Авторегрессионный процесс первого порядка: $y_t = c + a y_{t-1} + \varepsilon_t$,

где ε_t — ошибка

Авторегрессионный процесс порядка p (AR(*p*)-процесс): $y_t = c + \varepsilon_t + \sum_{i=1}^{p} a_i y_{t-i}$

Временной ряд t y называется *интегрированным порядка k* , k =1, 2... , если

- ряд y_t не является стационарным
- ряд $\Delta^k y$, полученный в результате k-кратного взятия разностей или дифференцирования ряда y_t, является стационарным рядом

- ряд $\Delta^{k-1}y$, полученный в результате (k –1)-кратного дифференцирования ряда y_t , не является стационарным рядом

Тест Дики-Фуллера (DF-тест) — это методика, которая используется для проверки временных рядов на стационарность. Является одним из тестов на единичные корни.

Временной ряд имеет единичный корень, или порядок интеграции один, если его первые разности образуют стационарный ряд. При помощи этого теста проверяют значение коэффициента a в авторегрессионном уравнении первого порядка : $y_t = ay_{t-1} + \varepsilon_t$. Если a = 1, то процесс имеет единичный корень, в этом случае ряд yt не стационарен, является интегрированным временным рядом первого порядка. Если |a| <1, то ряд стационарный.

Сущность DF-теста

Приведенное авторегрессионное уравнение AR(1) можно переписать в виде: $\Delta y_t = by_{t-1} + \varepsilon_t$,

где b = a−1, а Δ — оператор разности первого порядка

Поэтому проверка гипотезы о единичном корне в данном представлении означает проверку нулевой гипотезы о равенстве нулю коэффициента b.

Существует три версии теста:

1. Без константы и тренда
   $\Delta y_t = by_{t-1} + \varepsilon_t$

2. С константой, но без тренда:
   $\Delta y_t = b_0 + by_{t-1} + \varepsilon_t$

3. С константой и линейным трендом:
   $\Delta y_t = b_0 + b_1 t + by_{t-1} + \varepsilon_t$

Для каждого из трёх тестов существуют свои критические значения *DF*-статистики, которые берутся из специальной таблицы Дики-Фуллера. Если значение статистики лежит левее критического значения, то нулевая гипотеза о единичном корне отклоняется и процесс признается стационарным. В противном случае гипотеза не отвергается и процесс может содержать единичные корни, то есть быть нестационарным (интегрированным) временным рядом.

Для изучения внутренней структуры временного ряда применяется функция автокорреляции, которая представляет собой множество коэффициентов корреляции между временным рядом и этим же рядом, сдвинутым относительно первоначального положения на t моментов времени. Коэффициенты автокорреляции показывают связь между различными уровнями временного ряда.

Частичный коэффициент автокорреляции измеряет связь между текущим значением ряда и его предыдущими значениями. Совокупность частичных коэффициентов автокорреляции образует частичную автокорреляционную функцию.

В общем виде модель ARMA(p,q), где p – порядок авторегрессии, q – порядок скользящего среднего, выглядит следующим образом:

$$y_t = \alpha_1 y_{t-1} + \cdots + \alpha_p y_{t-p} - \theta_1 y_{t-1} - \cdots - \theta_q y_{t-q} + \varepsilon_t$$

Если процесс оказывается нестационарным и для приведения его к стационарному виду потребовалось взять несколько разностей, то модель становится ARIMA(p,d,q), где d – порядок разности.

Вклад:

Кирилл Куракин – описание программы в readme, выполнение 3 задания

Анастасия Коротчук – теоретическая часть readme, выполнение 1 и 2 задания

Библиотеки

- *Seaborne* — библиотека для визуализации данных, созданная на основе *matplotlib*. Предоставляет высокоуровневый интерфейс для создания информативных и красивых графиков. В нашей задаче *Seaborne* будет обозначаться как *sns*
- *Pandas* — программная библиотека для обработки и анализа данных, построенная поверх библиотеки *NumPy*. В нашей задаче для облегчения записи *Pandas* будет обозначаться как *pd*
- *Statsmodels* – модуль, предоставляющий доступ к классам и функциям для оценки множества разных математических моделей. В нашей задаче *Statmodels* будет обозначаться как *sm*
- *Sklearn* — библиотека, предоставляющая доступ к средствам прогнозирования
- *NumPy* — библиотека с открытым исходным кодом с такими возможностями, как поддержка многомерных массивов (включая матрицы), поддержка высокоуровневых математических функций, предназначенных для работы с многомерными массивами. В нашей задаче *NumPy* будет обозначаться как *np*.
- *Matplotlib* — визуализация решений. В нашей задаче для облегчения записи *Matplotlib* будет обозначаться как *plt*

Описание программы: ее ход и принцип действия, включая выводы

На вход программе подается таблица значений временного ряда формата *xlsx*

Откроем код – нас встретит *import* блок, в котором видны все библиотеки, которые мы используем в программе. Иногда, нам требуются лишь частные функции, а не вся библиотека – тогда мы импортируем исключительно их, например *from statsmodels.tsa.stattools import adfuller*.

Далее идут три внешние функции – кратко опишем, что они делают и как.

def test(data), def st(data, type) – очень похожие функции*,* проводящие тест Дики-Фуллера над рядом *data.* Первая проводит полный тест и выводит полную информацию о нем, вторая же выводит на экран краткое сообщение, в котором указывается тип ряда и стационарен ли он (*Y* – если да, *N* – если нет).

def plotseasonal(*res*, *axes*, *type*) необходима для построения красивых и наглядных графиков для двух моделей декомпозиции исходного ряда. Поскольку в программе будет строиться сетка *2*x*3*, то параметрами функции являются: *res* – результат разбиения, *axes* – номер колонки (задается в формате [:, *n*], потому что фиксируем мы номер колонки, а номер строки меняем по функции) и *type* – тип декомпозиции в виде строки. Тип нам необходим для задания названия кластера подграфиков – *axes*[*0*].*set_title*(*type*). Далее каждый из трех подграфиков мы по оси *Y* озаглавливаем соответствующе *axes*[*0*..*2*].*set_ylabel*('тип части декомп') ну и отрисовываем их последовательно: *res*.*seasonal*|*resid*|*trend*.*plot*(*ax* = *axes*[*0*..*2*], *legend* = *False*), где первый параметр – номер строки, где появится график, а второй задает отсутствие легенды графиков (чтобы картину не портило наименование линий).

Перейдем к основной части программы. Первыми шагами зададим:

- Путь к файлу с таблицей – в этой программе выбран способ с абсолютной ссылкой, потому как задача заключалась именно в анализе данного ряда, и смена ряда не предполагается. Выполняется командой *file = 'absolute link'*
- Размеры табличек – лист A*4*: *a4_dims* = (*x*,*y*)
- Задаем общие параметры для всех графиков при помощи средств *seaborne* - *sns*.*set*(*style*, *context*, *palette*, *rc*{'*figure*.*figsize*':}). Это стиль полотна для графиков, его вид, последовательность смены цветов линий и размер полотна соответственно.

Далее откроем файл с таблицей и переведем его в специальный тип из библиотеки *pandas* *DataFrame*, с которым дальше средствами этой самой библиотеки мы будем работать. Поскольку исходный файл *excel* воспользуемся функцией *dataset* = *readexcel*(ссылка на файл - *file*, название листа, с которого будет считан ряд - *sheet*, название колонки с датами – *parse_dates* = ['*Date*'], параметр, задающий значение дня как первое *dayfirst* = *True*, номер колонки, по которой ряд будет индексироваться *index_col* = *0*). Заметим, что в дальнейшем по *dataset*.*Value* мы будем обращаться именно к индексированному натуральными числами ряду значений.

После этого программа рисует график исходного ряда средствами *seaborne* и *matplotlib*: *sns*.*lineplot*(*data*, *linewidth*) – чертит двумерный график по данным *data* с шириной линий *linewidth*. Также в программе часто встречается *DataFrame*.*plot*() – встроенное в *pandas* средство быстрой печати графика по ряду. В целях экономии времени пользователя с этого момента подробных объяснений по печати графиков не будет.

Далее находятся два вида скользящих статистик для ряда – стандартное отклонение (дисперсия) и среднее, заданные в окне *10*, т.е. посчитанные локально для отрезка из *10* дат. Выбор такого окна обусловлен размерами данной нам статистики, и большие значения сильно смещают вправо по оси *X* в графическом представлении. В целях упрощения задачи, подсчитанные статистики присоединяются в виде дополнительных столбцов к *dataframe* без сбития индексирования. Реализуется это функциями *dataset*.*assign*(имя столбца*1* = столбец*1*, …) и *pd*.*Series*.*rolling*(ряд, окно = *10*).*std*() и .*mean*(), где последние две по заданному ряду и находят соответствующие статистики. После этого рисуется график по всем трем колонкам значений.

Посмотрим на этот график. Что нам, вообще говоря, от него нужно. Нашей целью в этом задании является прогнозирование. Зная определенный набор данных, мы хотим продолжить его, узнать примерную картину будущего. Для этого нам необходимо понять, с каким рядом мы имеем дело. Для стационарных рядов, то есть тех, что с течением времени не меняют своих статистических характеристик, задача прогнозирования представляется довольно простой, потому что можно предположить и дальнейшую неизменность его характеристик. Уже по одному виду исходного графика можно предположить, что он нестационарен – достаточно взглянуть на его довольно

быстрый рост. Такой же рост мы видим и для его средних значений, и хотя его дисперсия флактуирует в районе нулевых значений, можно предположить, что ряд нестационарен.

Следующий шаг – проведение теста Дики Фуллера на поиск единичных корней. Используется уже описанная внешняя функция *test*(*data*). Тест показывает, что они существуют, а значит ряд уже точно нестационарен. Но как нам с этим бороться? Можно попробовать провести прогнозирование не по самому ряду, а по его тренду. Для этого воспользуемся средствами сезонной декомпозиции библиотеки *statmodels.*

Функция *seasonal_decompose(dataset.Value, model = 'additive' // ' multiplicative')* проводит разбиение ряда одним из двух способов (заданным *model)* на *3* компоненты*:* случайную*,* сезонную (периодическую) и тренд, которые являются параметрами возвращаемого функцией значения. Было замечено, что при таком разбиении получившиеся компоненты обладают неприятным свойством – первые и последние *6* значений получившихся рядов (а точнее только случайного ряда и тренда) в представлении *DataFrame* являются *Nan*, что делает дальнейший анализ весьма затруднительным. Чтобы преодолеть эту проблему, программа обрезает эти первые и последние значения: *set.trend* // *resid* = *set.trend* // *resid.drop*(*index*[*0*..*6*] // *index*[*0*..*6*]), где // подразумеваются многоразовое выполнение.

После таких манипуляций программа выводит на экран в виде сетки *2*x*3* графики этих рядов. В правой колонке результаты аддитивного разбиения, в левой – мультипликативного. После этого проводится тест Дики-Фуллера для этих рядов. Поскольку нас интересует лишь вопрос их стационарности, то воспользуемся сокращенной формой теста *st*(*data*, *type*).

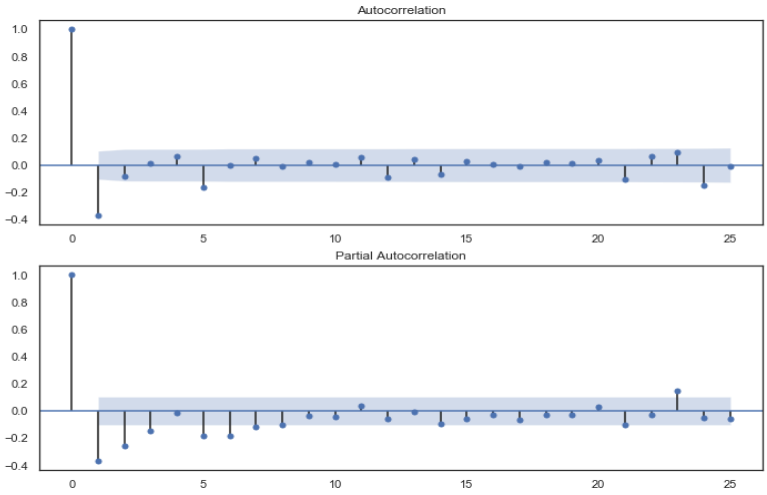
Посмотрим внимательно, на получившуюся картину. Имеет место годовая сезонность, однако, уже из периодичности следует то, что эта составляющая не влияет на стационарность исходного ряда в целом. И действительно, ряд сезонной компоненты стационарен как и ряд случайной. Из этого уже можно сделать вывод о нестационарности трендовой компоненты, что лишь подтверждается дальнейшим тестом.

Для прогнозирования этого ряда нам необходимо вычислить его порядок интегрирования. Для этого необходимо в цикле проводить тест Дики Фуллера, так как ряд является интегрированным первого порядка, если первыми разностями из него можно получить стационарный. В программе это реализуется циклом*:*

```
while i == 1:
   dftest = adfuller(differ)
   if dftest[0] <= dftest[4]['5%']:
   i = 0
   differ = differ.diff(periods=1).dropna()
   k = k + 1
if k:
    print("Order of integration of this timeset is ", k,'\n')
```

В цикле на каждом этапе проверяется наличие единичного корня, а после вычисляется разность. Подсчет начинается с *k* = *0*. Первая разность на момент начала уже высчитана. Таким образом, программа получает результат – ряд является интегрируемым порядка *1*.

Таким образом, мы получили первый параметр для создания модели *ARIMA*. Найдем остальные два – для этого построим графики автокорреляции и частичной автокорреляции для ряда первых разностей.



Верхний поможет нам определить параметр *p* – число автокорреляционных коэффициентов, сильно отличных от *0*, второй же поможет в определении *q*. Для этого нужно увидеть, что начиная с первого элемента все значения лагов отрицательны, а нулевой сильно отличается от нуля и только первый заметно выходит за синюю полосу (по условию *p* – максимальный номер сильно отличного от нуля коэффициента). Таким образом *p* = *1*, *q* = *1*.

Графики выводятся при помощи встроенной в *statmodels* функции *sm.graphics.tsa.plot_acf* // *pacf*(ряд, максимальное число лагов). Напомним, что лаг в статистике – временное запаздывание, а не сбой программы.

Теперь приступим непосредственно к созданию модели. Нам известны первые *360* значений ряда с разницей в значениях в месяц. Исходя из данных второй таблицы, для сравнения нужно спрогнозировать картину еще на *5* лет, т.е. привести еще *60* значений.

Для начала откроем заново первый файл в *df*, явно привязав значение дат к специальному типу Дата, при помощи *df.Date* = *pd.to_datetime*(*df.Date*). Также в *df1* откроем второй файл с результатами за следующие *5* лет.

В *model_arima* занесем модель *ARIMA* для нашего ряда с параметрами *p* = *1*, *q* = *1*, *d* = *1*, где *d* – порядок интегрирования: *model_arima* = *sm.tsa.ARIMA*(*df.Value*, *order* = (*1*,*1*,*1*)).*fit*(). В целях ознакомления распечатаем эту модель.

После этого начнем сам прогноз*: result = model_arima.predict(start = 360, end = 418, typ = 'levels').* Такими образом мы спрогнозируем результат, начиная с *360*-го индекса по *418*. Стоит отметить, что *ARIMA* – линейная модель, но не всегда излишняя точность дело полезное. Ее менее линейный собрат *SARIMA* требует куда более точные данные, сложнее настраивается и требует частого переобучения (хотя в нашей задаче это не проблема), а результат зачастую довольно похож на получаемый из модели *ARIMA*.

Остается только увидеть результат, но это задача осложнена сбитыми индексами – мы ведь прогнозировали на *5* лет, начиная с момента *360*. Предлагается весьма интересное решение – средствами *pandas* мы ряд, начиная с *360*-го элемента преобразуем в массив и обратно.

$$res1 = result.values;\ res2 = pd.DataFrame(res1)$$

Таким образом мы получаем ряд с *0*-го элемента из спрогнозированных значений.

Для наглядного сравнения рисуются два графика: на первом мы видим, как полученный прогноз продолжает исходный ряд. На втором мы можем сравнить прогноз и фактические данные. Однако визуального сравнения недостаточно – посчитаем коэффициент детерминации.

R2 = *r2_score*(*df1.value*[*1*:], *res2*), *r2_score* – функция библиотеки *sklearn.metrics*, считающая коэффициент детерминации для двух заданных рядом. Мы получили *r2* = *-3*. Отрицательные значения говорят о крайней неадекватности модели и действительно: общего между спрогнозированным рядом и реальными данными мало – разве что только оба ряда в целом растут.

Считается, что наилучшей будет модель с наименьшим значением критерия AIC, и в нашем случае это модель ARIMA с значением *248*. Последним штрихом выведем значения *r2* и *AIC* на экран.